Dimensional Model - SQL

Goal

To build a dimensional model for customer journeys (acquisition, expansion, churn), deriving key metrics (time-to-close, health scores, expansion sizing)

Datasets Used

The analysis leverages five interconnected CSV datasets, loaded into MySQL schema `mimecast` for modeling. Only relevant columns for lifecycle tracking (customer_id as join key, dates for temporal filters, metrics for KPIs) are used.

- customer_subscriptions.csv: Tracks subscription history, changes (e.g., plan_type, seats, monthly_value, start_date, end_date, status, change_reason).
Relevancy: churn events (status='Ended'), expansion (change_reason like 'Add Seats'), tenure (DATEDIFF(start_date, end_date)), and acquisition (first start_date where change_reason='New Customer').

- sales_activities.csv: Logs sales touchpoints (activity_date, activity_type like 'Demo'/'Closing Call', outcome, duration_minutes).
Relevancy:acquisition journey timing (pre-sub activities for time-to-close) and engagement proxies (positive outcomes count).

- support_tickets.csv: Records support interactions (created_date, ticket_type, resolution_hours, satisfaction_score).
Relevancy: health scores (avg satisfaction_score <4 signals risk) and churn indicators (high resolution_hours >48 or volume >3/month).

- product_usage.csv: Captures monthly feature adoption (month, feature, usage_percentage, login_frequency, feature_requests).
Relevancy: engagement in health scores (avg usage_percentage <50% as disengagement) and expansion signals (high feature_requests for upsell potential).

- marketing_campaigns.csv: Details lead attribution (campaign_date, response like 'Clicked', campaign_cost, attributed_revenue).
Relevancy: acquisition funnel (first campaign_date as lead proxy) and ROI (revenue/cost for quality leads).

## Data Preparation: Combining at Customer ID Level
Data is prepared via a star schema in MySQL: `dim_customer` (unique customer_id grain, ~500 rows from UNION DISTINCT across tables) serves as the central dimension, enriched with fact-level aggregates.
Preparation steps: (1) Load CSVs
(2) Create separate view at Customer_id level (customer_journey)
(3) Handle Nulls, Date issues in start and end date (string to date)
(4) Average out data by replacing nulls with mean value per customer

KPIs and Problems They Solve
The metrics address key questions: high-churn segments (health scores), fast closures (time-to-close), expansion potential (revenue sizing), support impact (via satisfaction in health), and contraction signals (net expansion <0).

<u>Customer acquisition journey (lead → trial → paid)</u>
- First Activity - For Customer Min(activity_date from sales)
- Acquisition Date - For Customer Min(start_date from subscription)
- Current Plan - For a customer latest Plan
- Status - For a customer latest Status

<u>Expansion events (seat adds, feature upgrades) and Churn Indicators</u>
- Upgrade: For Customer Count of Add Seats , Upgrade Plan
- Downgrade: For Customer Count of Drop Seats , Downgrade Plan
- Customer Health Scores (0-100 composite; buckets: Healthy >40, Monitor 20-40, At-Risk <20):
- avg_days_per_sub - For Customer average duration bw Subscription
- Engagement= (Avg Usage% + Avg Login/2) / 2 * 35%
- Tenure= Avg Days per Subscription/365 * 25%
- Status= Status score * 5%
- Net Expansion = Upgrade - Downgrade * 15%

<u>BPM</u>
- Time-to-Close for New Deals (avg DATEDIFF(first_activity, acquisition_date);
- HEALTH SCORE = 35% ENGAGEMENT + TENURE 25% + STATUS 5% + NET EXPANSION 15% + ROI 20%
- ROI = Revenue / Campaign Cost * 20%

AI Usage:

All tables are transaction tables there are no unique identifiers. Prompted AI to understand the flow of the problem statement.

```
Marketing Campaigns → Sales Activities → Subscriptions → Product Usage → Support Tickets
   ↓              ↓             ↓              ↓
 Lead        Acquisition    Expansion        Retention
```

List down the potential problems when we build relationships between tables

1. **Row explosion -** Joining multiple event-level tables on `customer_id` multiplies rows
   Solution: Aggregate first or use bridge table
2. **Ambiguous time overlap** - Support tickets from 2024 linked to 2022 subscription
   Solution: Use time-bound joins (`BETWEEN start_date AND end_date`)
3. **Inconsistent granularity -** Subscriptions = event-level, usage = monthly
   Solution: Normalize to a monthly snapshot
4. **Churn misclassification -** Active subscription may have ended but not updated yet
   Solution: Define clear churn logic (`status='Ended' AND end_date < current_date`)
5. **Expansion misdetection** - Add Seats vs Upgrade confusion
   Compare `LAG(seats)` and `LAG(plan_type)` per customer_id

Build Dimension and Fact tables Logics.

Cross validated the query with blockers like Lead dates have nulls, Sales activities has only 400 id but subscriptions has 500 ids

Generated KPI and use cases based on the data provided

sql

```
 View: mimecast.customer_journey
 Joins: LEFT on customer_id;
 Aggregates to customer level to handle many-to-one events.
 Output: ~500 rows;


CREATE OR REPLACE VIEW mimecast.customer_journey AS
WITH all_customers AS (
   SELECT customer_id FROM mimecast.customer_subscriptions
   UNION
   SELECT customer_id FROM mimecast.sales_activities
   UNION
   SELECT customer_id FROM mimecast.support_tickets
   UNION
   SELECT customer_id FROM mimecast.product_usage
   UNION
   SELECT customer_id FROM mimecast.marketing_campaigns
```

```
),
journey AS (
    SELECT
        s.customer_id,
        MIN(s.activity_date) AS first_activity,
        COUNT(DISTINCT s.activity_id) AS total_sales_activities
    FROM mimecast.sales_activities s
    JOIN mimecast.customer_subscriptions m
        ON s.customer_id = m.customer_id
    WHERE s.activity_date < (
        SELECT MIN(m2.start_date)
        FROM mimecast.customer_subscriptions m2
        WHERE m2.customer_id = s.customer_id
    )
    GROUP BY s.customer_id
),

latest_status AS (
    SELECT
        customer_id,
        status
    FROM (
        SELECT
            *,
            ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY start_date DESC, subscription_id DESC) AS rn
        FROM mimecast.customer_subscriptions
    ) ranked
    WHERE rn = 1
),
plan AS (
    SELECT
        cs.customer_id,
        cs.plan_type
    FROM mimecast.customer_subscriptions cs
    JOIN (
        SELECT
            customer_id,
            MAX(start_date) AS latest_start_date
        FROM mimecast.customer_subscriptions
        GROUP BY customer_id
    ) recent
      ON cs.customer_id = recent.customer_id
     AND cs.start_date = recent.latest_start_date
),
marketing AS (

    SELECT
        customer_id,
        COUNT(DISTINCT campaign_id) AS total_campaigns_exposed,
        ROUND(
            SUM(attributed_revenue) *100/ NULLIF(SUM(campaign_cost), 0),
            0
        ) AS ROI_per,

           ROUND(
            SUM(CASE WHEN conversion = "TRUE" THEN 1 ELSE 0 END) * 100.0 / COUNT(*),
            0
        )

         AS conversion_rate_per
    FROM mimecast.marketing_campaigns
    GROUP BY customer_id

),
-- Inline dim_customer logic
dim AS (
    SELECT
        ac.customer_id,
        j.first_activity AS lead_date,
        MIN(cs.start_date) AS acquisition_date,
        j.total_sales_activities,
        p.plan_type AS current_plan_type,
        COUNT(DISTINCT cs.plan_type) AS plan_used_tilldate,
        ls.status AS current_status,
        SUM(CASE
                WHEN UPPER(cs.change_reason) IN ('ADD SEATS', 'UPGRADE PLAN') THEN 1
                ELSE 0
            END) AS upgraded_tilldate,
        SUM(CASE
                WHEN UPPER(cs.change_reason) IN ('REMOVE SEATS', 'DOWNGRADE PLAN') THEN 1
                ELSE 0
            END) AS downgraded_tilldate,
        MAX(cs.start_date) AS last_sub_activated,

        COUNT(DISTINCT cs.subscription_id) AS subscription_count
 FROM all_customers ac
    LEFT JOIN mimecast.customer_subscriptions cs ON ac.customer_id = cs.customer_id
    LEFT JOIN journey j ON ac.customer_id = j.customer_id
    LEFT JOIN latest_status ls ON ac.customer_id = ls.customer_id
    LEFT JOIN plan p ON ac.customer_id = p.customer_id
    GROUP BY ac.customer_id, ls.status, p.plan_type, j.first_activity, j.total_sales_activities
),
-- satisfaction AS (
   -- SELECT
     --   customer_id,
       -- COALESCE(AVG(satisfaction_score), 1.98) AS avg_satisfaction
   -- FROM mimecast.support_tickets
   -- GROUP BY customer_id
-- ),
engagement AS (
    SELECT
        customer_id,
        ROUND(COALESCE(AVG(COALESCE(usage_percentage, 0)), 50.0), 2) AS avg_usage_pct,
        round(COALESCE(AVG(COALESCE(login_frequency,0)), 10.0),2) AS avg_login_freq
    FROM mimecast.product_usage
    GROUP BY customer_id
),
tenure_status AS (
    SELECT
        cs.customer_id,
        ROUND(AVG(
            DATEDIFF(
                CASE
                    WHEN cs.status = 'Active' THEN CURDATE()
                    WHEN cs.status = 'Churned' THEN cs.start_date
                    ELSE cs.end_date
                END,
                cs.start_date
            )
        ) ) AS avg_days_per_sub,
        MAX(CASE WHEN status = 'Active' THEN 100 ELSE 0 END) AS status_score,
        SUM(CASE
                WHEN UPPER(cs.change_reason) IN ('ADD SEATS', 'UPGRADE PLAN') THEN 1
                ELSE 0
            END) AS upgraded_tilldate,
        SUM(CASE
                WHEN UPPER(cs.change_reason) IN ('REMOVE SEATS', 'DOWNGRADE PLAN') THEN 1
                ELSE 0
            END) AS downgraded_tilldate
    FROM mimecast.customer_subscriptions cs

    GROUP BY cs.customer_id
)

SELECT
    d.customer_id,
    d.lead_date,
    d.acquisition_date,
    d.total_sales_activities,
    d.current_plan_type,
    d.plan_used_tilldate,
    d.current_status,
    d.subscription_count,
    ts.upgraded_tilldate,
    ts.downgraded_tilldate,
   d.last_sub_activated,
   -- marketing_campaigns
   mc.total_campaigns_exposed,
    mc.ROI_per,
    mc.conversion_rate_per,
    -- Health score inputs
    -- s.avg_satisfaction AS ticket_satisfaction,
    e.avg_usage_pct,
    e.avg_login_freq,
    ts.avg_days_per_sub,
    ts.status_score,
    -- Final health score
ROUND(
    LEAST(
        (
            (
                (COALESCE(e.avg_usage_pct, 0) + (COALESCE(e.avg_login_freq, 0) / 2)) / 2 * 0.35
            ) +                                                    -- Engagement (35%)
            (LEAST(COALESCE(ts.avg_days_per_sub, 0) / 365, 1) * 25) +         -- Tenure (25%)
            (COALESCE(ts.status_score, 0) * 0.05) +                            -- Status (5%)
            (
                LEAST(
                    GREATEST(
                        COALESCE(ts.upgraded_tilldate, 0) - COALESCE(ts.downgraded_tilldate, 0),
                        0
                    ),
                    10
                ) * 1.5
            ) +                                                    -- Net Expansion (15%)
            (LEAST(COALESCE(mc.ROI_per, 0) / 100, 1) * 20)                     -- ROI (20%)
        ),
        100
    ),
    2
) AS health_score,

CASE
  when   d.current_status = "Churned" then "Churned"
```

```
    WHEN (
        ROUND(
            LEAST(
                (
                    (
                        (COALESCE(e.avg_usage_pct, 0) + (COALESCE(e.avg_login_freq, 0) / 2)) / 2 * 0.35
                    ) +
                    (LEAST(COALESCE(ts.avg_days_per_sub, 0) / 365, 1) * 25) +
                    (COALESCE(ts.status_score, 0) * 0.05) +
                    (
                        LEAST(
                            GREATEST(
                                COALESCE(ts.upgraded_tilldate, 0) - COALESCE(ts.downgraded_tilldate, 0),
                                0
                            ),
                            10
                        ) * 1.5
                    ) +
                    (LEAST(COALESCE(mc.ROI_per, 0) / 100, 1) * 20)
                ),
                100
            ),
            2
        )
    ) > 40 THEN 'Healthy'
    WHEN (
        ROUND(
            LEAST(
                (
                    (
                        (COALESCE(e.avg_usage_pct, 0) + (COALESCE(e.avg_login_freq, 0) / 2)) / 2 * 0.35
                    ) +
                    (LEAST(COALESCE(ts.avg_days_per_sub, 0) / 365, 1) * 25) +
                    (COALESCE(ts.status_score, 0) * 0.05) +
                    (
                        LEAST(
                            GREATEST(
                                COALESCE(ts.upgraded_tilldate, 0) - COALESCE(ts.downgraded_tilldate, 0),
                                0
                            ),
                            10
                        ) * 1.5
                    ) +
                    (LEAST(COALESCE(mc.ROI_per, 0) / 100, 1) * 20)
                ),
                100
            ),
            2
        )
    ) > 20 THEN 'Monitor'
    ELSE 'At-Risk'
END AS health_bucket



FROM dim d
-- LEFT JOIN satisfaction s ON d.customer_id = s.customer_id
LEFT JOIN engagement e ON d.customer_id = e.customer_id
LEFT JOIN tenure_status ts ON d.customer_id = ts.customer_id
LEFT JOIN marketing mc ON d.customer_id = mc.customer_id;
```